

# Two-Dimensional Array

(อาร์เรย์ 2 มิติ)

ทัดดาว ปานสมบัติ

คณะวิทยาศาสตร์และเทคโนโลยี

มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตปัตตานี

(Last updated on 25 มีนาคม 2565 เวลา 8.00 น.)

# Introduction of 2D Array

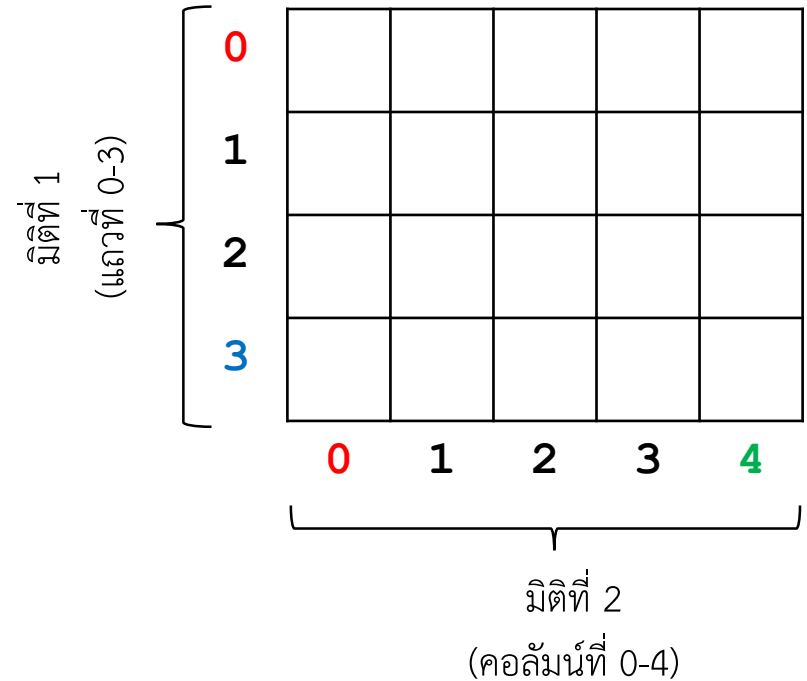
# ตัวแปรอาร์เรย์ 2 มิติ

ตัวแปรอาร์เรย์ 2 มิติ

- เก็บข้อมูลชนิดเดียวกันในลักษณะ 2 มิติ
- มองได้เป็นตารางที่มีข้อมูลในแนวแถว (row) และคอลัมน์ (column)

ตัวแปรอาร์เรย์ 2 มิติ ขนาด  $M \times N$

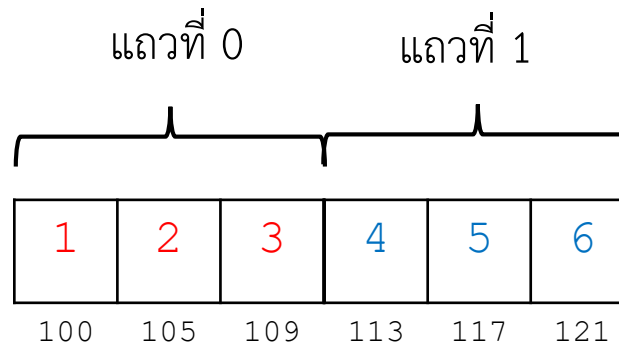
- แถวเริ่มจากแถวที่ 0 ถึงแถวที่  $M - 1$
- คอลัมน์เริ่มจากคอลัมน์ที่ 0 ถึงคอลัมน์  $N - 1$
- เก็บข้อมูลได้  $M \times N$  ตัว



อาร์เรย์ 2 มิติขนาด  $4 \times 5$

# การเก็บอาร์เรย์ 2 มิติ ในหน่วยความจำ

0	1	2	3
1	4	5	6
	0	1	2



↳ เซลล์แบบ ๒๒ มี ๑๐๓

อาร์เรย์ 2 มิติขนาด  $2 \times 3$

ในหน่วยความจำ

# การประกาศตัวแปรอาร์เรย์ 2 มิติ

```
type arrayName [ rows ] [ columns ]
```

**type** คือ ชนิดของข้อมูลที่จะเก็บในตัวแปรอาร์เรย์

**arrayName** คือ ชื่อตัวแปรอาร์เรย์

**rows** คือ จำนวนสูงสุดของข้อมูลที่สามารถเก็บในแถวของอาร์เรย์

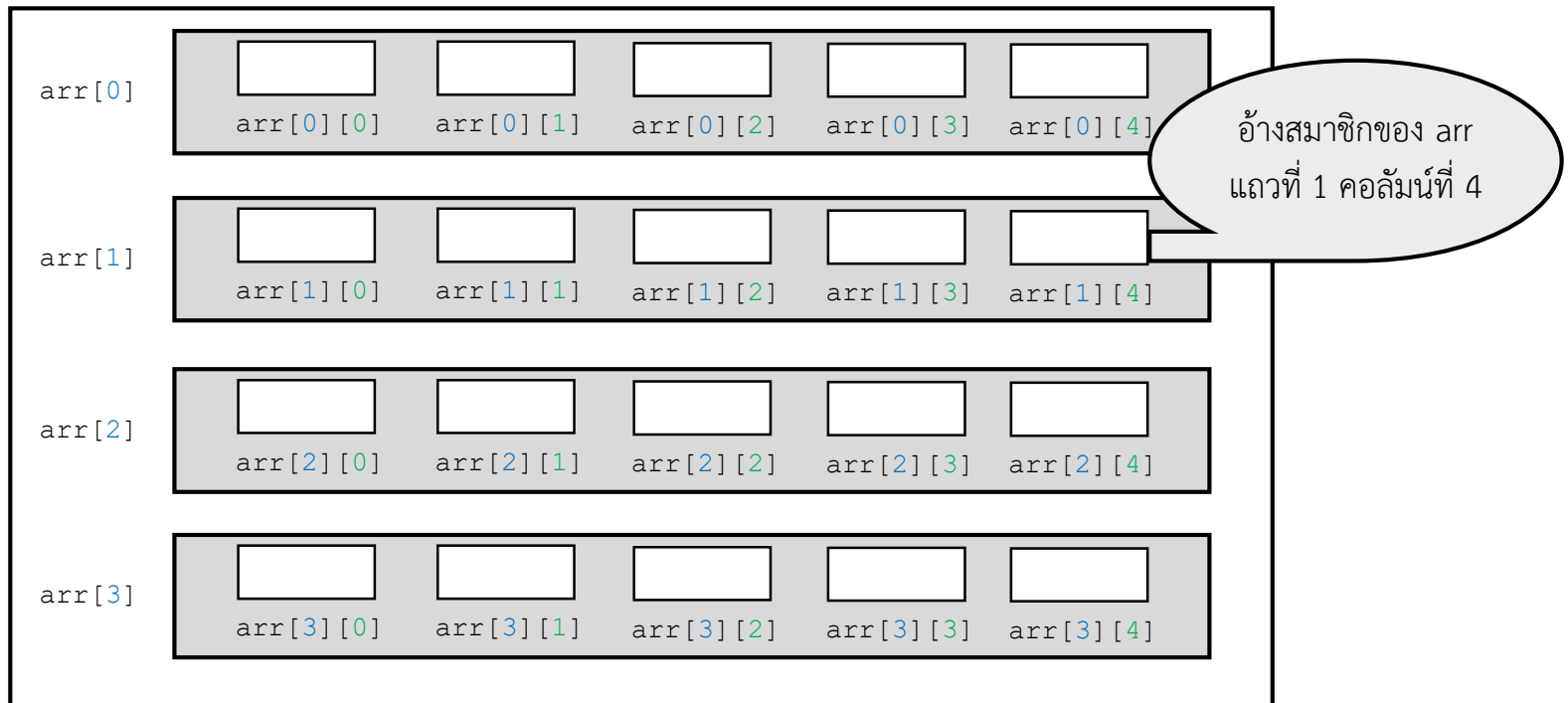
**columns** คือ จำนวนสูงสุดของข้อมูลที่สามารถเก็บในแนวคอลัมน์ของอาร์เรย์

ตัวอย่างเช่น

```
int arr[4][5];  
float numbers[10][15];  
char data[6][6];
```

# การเก็บค่าและการอ้างอิงถึงค่าในอาร์เรย์ 2 มิติ

```
int arr[4][5];
```



# การกำหนดค่าเริ่มต้น (Array Initialization)



แบบระบุขนาดของแถวและคอลัมน์

```
int arr[3][4] = { {0, 1, 2, 3},  
                  {10, 11, 12, 13},  
                  {20, 21, 22, 23} } ;
```

```
int arr[3][4] = {0, 1, 2, 3,  
                 10, 11, 12, 13,  
                 20, 21, 22, 23};
```

0	0	1	2	3
1	10	11	12	13
2	20	21	22	23
	0	1	2	3



แบบไม่ระบุขนาดของแถว แต่ระบุขนาดคอลัมน์

```
int arr[][4] = { {0, 1, 2, 3},  
                 {10, 11, 12, 13},  
                 {20, 21, 22, 23} } ;
```

```
int arr[][4] = {0, 1, 2, 3,  
                 10, 11, 12, 13,  
                 20, 21, 22, 23};
```

# การกำหนดค่าเริ่มต้น (Array Initialization)

แบบระบุขนาดของแถว แต่ไม่ระบุขนาดคอลัมน์ => **compile error**



```
int arr[3][] = { {0, 1, 2, 3},  
                 {10, 11, 12, 13},  
                 {20, 21, 22, 23} } ;  
  
int arr[3][] = {0, 1, 2, 3,  
                10, 11, 12, 13,  
                20, 21, 22, 23};
```

แบบไม่ระบุขนาดของแถว และไม่ระบุขนาดของคอลัมน์ => **compile error**

สรุปว่าต้อง  
ระบุขนาด  
คอลัมน์



```
int arr[][] = { {0, 1, 2, 3},  
                {10, 11, 12, 13},  
                {20, 21, 22, 23} } ;  
  
int arr[][] = {0, 1, 2, 3,  
               10, 11, 12, 13,  
               20, 21, 22, 23};
```



# การกำหนดค่าเริ่มต้นเป็นค่าที่เหมือนกันทุกค่า

กำหนดให้ทุกค่าเป็น 0

```
int arr[3][4] = {} ;  
int arr[3][4] = {0} ;
```

0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
	0	1	2	3

กำหนดให้ทุกค่าเป็นค่าอื่นที่ไม่ใช่ 0

```
int arr[3][4];  
for(r = 0; r < 3; r++)  
    for(c = 0; c < 4; c++)  
        arr[r][c] = 2;
```

0	2	2	2	2
1	2	2	2	2
2	2	2	2	2
	0	1	2	3

# การเข้าถึงสมาชิกในอาร์เรย์ 2 มิติ

`arrayName` [ `rowIndex` ] [ `colIndex` ]

`arrayName` คือ ชื่อตัวแปรอาร์เรย์

`rowIndex` คือ ค่าดัชนีกำกับในแนวแถว

`colIndex` คือ ค่าดัชนีกำกับในแนวคอลัมน์

ตัวอย่างเช่น

```
int arr[3][2] = {{1, 2}, {3, 4}, {5, 6}};  
arr[0][1] = 50;    // {{1, 50}, {3, 4}, {5, 6}}  
arr[2][1] = arr[1][1] + 10; // {{1, 50}, {3, 4}, {5, 14}}
```

# การรับค่าด้วย `scanf` และ

# การแสดงผลด้วย `printf`

## การรับค่าด้วย `scanf`

```
scanf("%d", &arr[0][0]); // arr is an array of int
scanf("%d", &arr[i][j]);
scanf("%f", &num[0][0]); // num is an array of float
scanf("%f", &num[i][j]);
```

## การแสดงผลด้วย `printf`

```
printf("%d", arr[0][0]); // arr is an array of int
printf("%d", arr[i][j]);
printf("%f", num[0][0]); // num is an array of float
printf("%.2f", num[i][j]);
```

# การรับขนาดของอาร์เรย์ด้วย `scanf` และ

```
#include <stdio.h>
int main(){
    int M, N, r, c;
    scanf("%d%d", &M, &N);
    int arr[M][N];          // cannot do int arr[M][N] = {0};
    for (r = 0; r < M; r++){
        for (c = 0; c < N; c++){
            printf("%d ", arr[r][c]);
        }
        printf("\n");
    }
    return 0;
}
```

ผลลัพธ์

**3 3**

**3 0 4205650**

**0 4210688 0**

**6421992 0 6421996**

# การกำหนดขนาดของอาร์เรย์

๖๖ ลัว ๖๖๑ ๖๖๑

```
#include <stdio.h>
#define M 3
#define N 2
int main(){
    int r, c;
    int arr[M][N];
    for (r = 0; r < M; r++){
        for (c = 0; c < N; c++){
            printf("%d ", arr[r][c]);
            printf("\n");
        }
    }
    return 0;
}
```

```
#include <stdio.h>
int main(){
    int r, c;
    const int M = 3;
    const int N = 2;
    int arr[M][N];
    for (r = 0; r < M; r++){
        for (c = 0; c < N; c++){
            printf("%d ", arr[r][c]);
            printf("\n");
        }
    }
    return 0;
}
```

ใช้ preprocessor directive #define

- เพิ่มเวลาในการ compile
- ลดหน่วยความจำที่ใช้
- เป็นการแทนค่า M ด้วย 3 และ N ด้วย 2 ทุกที่ใน source code แล้วนำไป compile

ใช้ตัวแปร

- กำหนดเป็น const int หรือ int ก็ได้
- เพิ่มหน่วยความจำที่ใช้
- ทราบชนิดข้อมูลของตัวแปร M และ N
- ใช้ได้กับมาตรฐาน C99 เป็นต้นไป

# Finding, Summing & Counting Elements in 2D Array

get

# หาผลรวมของค่าในอาร์เรย์ 2 มิติ

```
#include <stdio.h>
#define M 5
#define N 3
int main(){
    int r, c, sum = 0;
    int arr[M][N] = {1,2,3,
                      1,2,3,
                      1,2,3,
                      1,2,3,
                      1,2,3};

    for (r = 0; r < M; r++)           // for each row
        for (c = 0; c < N; c++)       // for each column
            sum += arr[r][c];
    printf("%d\n", sum);
    return 0;
}
```

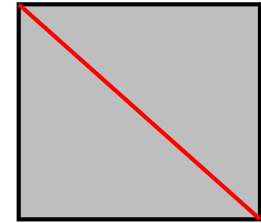
ผลลัพธ์

30

1,1)

get

# หาผลรวมของค่าบนเส้นทแยงมุม



```
#include <stdio.h>
#define N 5
int main(){
    int r, sum = 0;
    int arr[N][N] = {1,1,2,0,0,
                     1,1,1,1,1,
                     0,0,0,2,2,
                     1,1,1,1,1,
                     0,0,0,0,0};
    for (r = 0; r < N; r++){
        sum += arr[r][r];
    }
    printf("%d\n", sum);
    return 0;
}
```

ผลลัพธ์

3

(0,0)

(1,1)

...

(N,N)

sum = 0;

```
for(int i=0; i<n; i++){
    sum += arr[i][i];
}
```

cout &lt;&lt; sum;

บนเส้นทแยงมุม ดัชนีของแถวเท่ากับดัชนีของคอลัมน์



get  
in the pocket

# หาผลรวมของค่าที่อยู่รอบขอบอาร์เรย์

```
#include <stdio.h>
```

```
#define M 5
```

```
#define N 3
```

```
int main(){
```

```
    int r, c, sum = 0;
```

```
    int arr[M][N] = {1,2,3,
```

```
                    1,2,3,
```

```
                    1,2,3,
```

```
                    1,2,3,
```

```
                    1,2,3};
```

```
    for (c = 0; c < N; c++)
```

```
        sum += arr[0][c] + arr[M-1][c];
```

```
    for (r = 1; r <= M-2; r++)
```

```
        sum += arr[r][0] + arr[r][N-1];
```

```
    printf("%d\n", sum);
```

```
    return 0;
```

```
}
```

บน  
i 0,1,2  
j 0,1,2

ล่าง  
i 4  
j 0,1,2

int c;

for (c = 0; c < N; c++)

sum += arr[0][c] + arr[M-1][c];

for (r

// upper and lower

// left and right

ผลลัพธ์

24

หัว

(1,0)

(2,0)

(3,0)

หาง

(1,3)

(2,3)

(3,3)

for (r = 0; r < M-2; r++)  
sum += arr[r][0] + arr[r][N-1];

เส้นทแยงมุมนี้ดัชนีของแถวเท่ากับดัชนีของคอลัมน์

# หาค่ามากที่สุดของแต่ละคอลัมน์ ในอาร์เรย์ 2 มิติ

no get

```
#include <stdio.h>
#define M 4
#define N 3
int main(){
    int r, c, maximum;
    int arr[M][N] = {1, 2, 3,
                     2, 7, 5,
                     9, 2, 5,
                     7, 8, 0};
    for (c = 0; c < N; c++){ // for each column
        maximum = arr[0][c];
        for (r = 1; r < M; r++) // for each row
            if (arr[r][c] > maximum)
                maximum = arr[r][c];
        printf("column %d has maximum = %d\n", c, maximum);
    }
    return 0;
}
```

ผลลัพธ์

column 0 has maximum = 9  
column 1 has maximum = 8  
column 2 has maximum = 5

```
#include <stdio.h>
```

```
#define M 4
```

```
#define N 3
```

```
int main(){
```

```
    int r, c, maximum;
```

```
    int arr[M][N] = { 1, 2, 3,  
                       2, 7, 5,  
                       9, 2, 5,  
                       8, 8, 0};
```

```
    for( c=0; c < N; c++) {  
        maximum = arr[0][c] ← เริ่มต้น
```

```
        for( r=1; r < M; r++) {  
            if( arr[r][c] > maximum) { ← เปรียบเทียบ  
                maximum = arr[r][c];  
            }  
        }  
        cout << maximum << "\n";
```

```
    return
```

```
}
```

# นับจำนวนสมาชิกบางค่าในอาร์เรย์ 2 มิติ

```
#include <stdio.h>
#define M 5
#define N 5
int main(){
    int r, c, count = 0;
    int x = 0;          //element for counting
    int arr[M][N] = {1,1,2,0,0,
                     1,1,1,1,1,
                     0,0,0,2,2,
                     1,1,1,1,1,
                     0,0,0,0,0};

    for (r = 0; r < M; r++)
        for (c = 0; c < N; c++)
            if(arr[r][c] == x)
                count++;

    printf("%d\n", count);
    return 0;
}
```

ผลลัพธ์

10

# Application of 2D Array with Matrix Representation & Operations

get

# การบวกของ Matrix

$$\begin{matrix} A & + & B & = & C \\ \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & a_{2,2} \end{bmatrix}_{3 \times 3} & + & \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} \\ b_{1,0} & b_{1,1} & b_{1,2} \\ b_{2,0} & b_{2,1} & b_{2,2} \end{bmatrix}_{3 \times 3} & = & \begin{bmatrix} a_{0,0} + b_{0,0} & a_{0,1} + b_{0,1} & a_{0,2} + b_{0,2} \\ a_{1,0} + b_{1,0} & a_{1,1} + b_{1,1} & a_{1,2} + b_{1,2} \\ a_{2,0} + b_{2,0} & a_{2,1} + b_{2,1} & a_{2,2} + b_{2,2} \end{bmatrix}_{3 \times 3} \end{matrix}$$

```
#include <stdio.h>
#define ROWS 3
#define COLS 3
int main(){
    int r,c;
    int A[ROWS][COLS] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    int B[ROWS][COLS] = {1, 1, 1, 1, 1, 1, 1, 1, 1};
    int C[ROWS][COLS];
    for (r = 0; r < ROWS; r++)
        for (c = 0; c < COLS; c++)
            C[r][c] = A[r][c] + B[r][c];
    return 0;
}
```

ค่าของอาร์เรย์ C

2	3	4
5	6	7
8	9	10



→ ลองไปเขียน

# การคูณของ Matrix

$$\begin{matrix} A & \times & B & = & C \\ \begin{bmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & a_{1,1} \\ a_{2,0} & a_{2,1} \end{bmatrix} & \times & \begin{bmatrix} b_{0,0} & b_{0,1} \\ b_{1,0} & b_{1,1} \end{bmatrix} & = & \begin{bmatrix} a_{0,0}b_{0,0} + a_{0,1}b_{1,0} & a_{0,0}b_{0,1} + a_{0,1}b_{1,1} \\ a_{1,0}b_{0,0} + a_{1,1}b_{1,0} & a_{1,0}b_{0,1} + a_{1,1}b_{1,1} \\ a_{2,0}b_{0,0} + a_{2,1}b_{1,0} & a_{2,0}b_{0,1} + a_{2,1}b_{1,1} \end{bmatrix} \end{matrix}$$

*Handwritten notes: Red circles around  $a_{0,0}$ ,  $a_{0,1}$ ,  $b_{0,0}$ ,  $b_{1,0}$ , and the first row of the result matrix. Blue annotations show dimensions:  $3 \times 2$  for A,  $2 \times 2$  for B, and  $3 \times 2$  for C. A blue box highlights the indices  $i, j$  in the first row of A.*

```
#include <stdio.h>
#define A_ROWS 3
#define A_COLS 2
#define B_ROWS 2
#define B_COLS 2
int main(){
    int i;    // i is index of a row in A
    int j;    // j is index of a column in A and a row in B
    int k;    // k is index of a column in B
    int sum;
    int A[A_ROWS][A_COLS] = {1, 2, 3, 4, 5, 6};
    int B[B_ROWS][B_COLS] = {1, 2, 2, 1};
    int C[A_ROWS][B_COLS];
    for (i = 0; i < A_ROWS; i++){
        for (k = 0; k < B_COLS; k++){
            sum = 0;
            for (j = 0; j < A_COLS; j++){
                sum += A[i][j] * B[j][k];
            }
            C[i][k] = sum;
        }
    }
    return 0;
}
```

*Handwritten notes: Blue annotations show dimensions: 3 for A\_ROWS, 2 for A\_COLS, 2 for B\_ROWS, 2 for B\_COLS. Blue numbers 1, 2, 3, 4, 5, 6 are written next to the first row of A. Blue numbers 1, 2, 2, 1 are written next to the first row of B. Blue numbers 0, 0, 0, 0 are written next to the first row of C. Blue numbers 1, 1 are written next to the second row of C.*

ค่าของอาร์เรย์ C

5	4
11	10
17	16



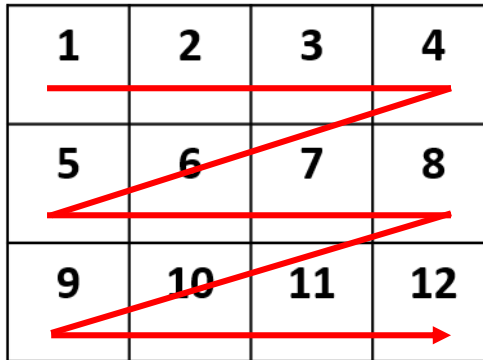
# A Walk in 2D Array

# Row-major order

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

start

1	2	3	4
5	6	7	8
9	10	11	12



```
#include <stdio.h>
#define ROWS 3
#define COLS 4
int main(){
    int r,c;
    int arr[ROWS][COLS] = {1, 2, 3, 4,
                           5, 6, 7, 8,
                           9, 10, 11, 12};
    for (r = 0; r < ROWS; r++)          // row
        for (c = 0; c < COLS; c++)      // column
            printf("%d ", arr[r][c]);

    return 0;
}
```

# Column-major order

1, 5, 9, 2, 6, 10, 3, 7, 11, 4, 8, 12

start

1	2	3	4
5	6	7	8
9	10	11	12

```
#include <stdio.h>
#define ROWS 3
#define COLS 4
int main(){
    int r,c;
    int arr[ROWS][COLS] = {1, 2, 3, 4,
                           5, 6, 7, 8,
                           9, 10, 11, 12};

    for (c = 0; c < COLS; c++)          // column
        for (r = 0; r < ROWS; r++)      // row
            printf("%d ", arr[r][c]);

    return 0;
}
```

→ ตามเข็มนาฬิกา

# Spiral order



1, 2, 3, 4, 8, 12, 11, 10, 9, 5, 6, 7

start

1	2	3	4
5	6	7	8
9	10	11	12

```
#include <stdio.h>
int main()
{
    int M, N, r, c;
    int rb = 0, rn;
    int cb = 0, cn;
    scanf("%d%d", &M, &N);
    int arr[M][N];
    rn = M;
    cn = N;
    for (r = 0; r < M; r++)
        for (c = 0; c < N; c++)
            scanf("%d", &arr[r][c]);
}
```

```

while (rb < rn && cb < cn){
    // print top unprinted row (left->right)
    for (c = cb; c < cn; c++)
        printf("%d ", arr[rb][c]);
    rb++;
    // print right unprinted col (top->bottom)
    for (r = rb; r < rn; r++)
        printf("%d ", arr[r][cn - 1]);
    cn--;
    // print bottom unprinted row (right->left)
    if(rb < rn){
        for (c = cn - 1; c >= cb; c--)
            printf("%d ", arr[rn - 1][c]);
        rn--;
    }
    // print left unprinted column (bottom->top)
    if(cb < cn){
        for (r = rn - 1; r >= rb; r--)
            printf("%d ", arr[r][cb]);
        cb++;
    }
}
}

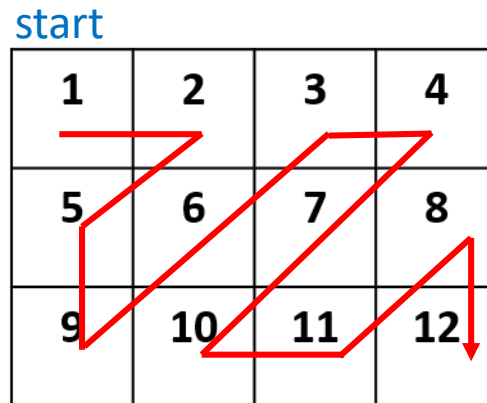
```



start

1	2	3	4
5	6	7	8
9	10	11	12

# Zig-Zag



1, 2, 5, 9, 6, 3, 4, 7, 10, 11, 8, 12

# Checking elements around an 2D Array's element

# เทคนิคการใช้อาร์เรย์ 1 มิติเป็น mask

อาร์เรย์ 7

r-1	0	1	2
r	3	x	4
r+1	5	6	7
	c-1	c	c+1

พิจารณา 8 ทิศรอบ ๆ

แถว → `int dr[] = { -1, -1, -1, 0, 0, 1, 1, 1 };;`  
 แนวก → `int dc[] = { -1, 0, 1, -1, 1, -1, 0, 1 };;`

r-1		0	
r	1	x	2
r+1		3	
	c-1	c	c+1

พิจารณา 4 ทิศรอบ ๆ

```
int dr[] = { -1, 0, 0, 1 };;
int dc[] = { 0, -1, 1, 0 };;
```



1	1	1	1	1
1	1	1	1	1
0	1	1	1	1
1	1	1	1	0
0	0	1	1	1

แสดง Index ของสมาชิกที่  
ล้อมรอบด้วย 1 ทั้ง 8 ทิศ

gg check

```
#include <stdio.h>
int main()
{
    /*  0 1 2
        4   5
        6 7 8   */
    int M, N, r, c, i, sum;
    int dr[] = {-1, -1, -1, 0, 0, 1, 1, 1};
    int dc[] = {-1, 0, 1, -1, 1, -1, 0, 1};
    scanf("%d%d", &M, &N);
    int arr[M][N];
    for (r = 0; r < M; r++)
        for (c = 0; c < N; c++)
            scanf("%d", &arr[r][c]);

    for(r = 1; r < M - 1; r++){
        for(c = 1; c < N - 1; c++){
            sum = 0;
            for(i = 0; i < 8; i++)
                if (arr[r + dr[i]][c + dc[i]] == 1) sum++;
            if(sum == 8) printf("(%d, %d)\n", r, c);
        }
    }
    return 0;
}
```